Introduktionslaboration för IP

Eric Elfving
eric.elfving@liu.se

25 augusti 2013

1 Introduktion

Denna introduktionslaboration är till för att du ska lära dig hantera systemet på ett bättre sätt med hjälp av inbyggda kommandon i terminalen. Först beskrivs uppgiften därefter kommer beskrivningar och tips på kommandon som kan vara bra att ha för att lösa uppgiften. Förutom det som finns i denna guide rekommenderas läsning och egna övningar i STONE (`http://www.ida.liu.se/stone/main`) samt läsning av respektive kommandos manualsidor med hjälp av kommandot `man`.

2 Uppgift

På filen `http://www.ida.liu.se/~TDP001/material/resultat.txt` finns den datamängd som ska behandlas. Varje person i denna fil har ett antal heltal som representerar deras poäng för tre olika uppgifter. Din uppgift är att summera dessa poäng och skriva ut resultatet sorterat i sjunkande ordning på formatet `<poängsumma> <namn>`. För den givna filen bör alltså resultatet bli detta:

```
50 Göran Johansson
48 Josefine Carlson
  ... <några rader borttagna av utrymmesmässiga själ>
23 Britt Lidell
22 Klas Hägglund
11 Jerker Leo
```

3 Terminalkommandon

Terminalen är ett verktyg som alla UNIX/LINUX-användare bör vara väl förtrogna med. Ska man vara riktigt noggrann finns det program som heter terminalemulatorer och det är dessa vi använder oss av. I Ubuntu (och många andra LINUX-distributioner) används vanligen `BASH` som står för GNU Bourne-Again SHell som är byggd för att ersätta den gamla Bourne shell (sh). Grunden i alla terminalkommandon är att ett program startas i terminalen, får indata från användaren via tangentbordet (standard input eller STDIN) och skriver ut sitt resultat i terminalförnstret igen (standard output, STDOUT). I följande del beskrivs hur man kan styra inmatning och utskrift med hjälp av terminalen.

3.1 BASH

BASH är ett väldigt stort och omfattande verktyg med massor av egenskaper. I detta kapitel beskrivs några få viktiga delar men det rekommenderas starkt att läsa på och öva på egen hand. Manualsidorna är väldigt utförliga.

3.1.1 Starta program

För att starta ett program i terminalen skriver man vanligen endast programmets namn. För att styra specifika inställningar för programmet kan man använda sig av så kallade väljare (eller flaggor, options). Vilka väljare ett visst program har kan man oftast få reda på med väljaren `-h` eller programmets manualsidor som man kan hitta med programmet `man`. Testa programmet man genom att ta reda på mer information om man:

```
man man
```

3.1.2 Omdirigering av in- och utmatning

Ett program som startas i terminalen får sin indata från stdin och skriver ut resultatet på stdout. Detta är dock inte alltid optimalt och därför finns omdirigering. De två enklaste fallen av omdirigering är att skriva ut på eller läsa in från filer. Det kan man göra med hjälp av operatorerna > och <. Följande kommando skriver ut "Hejsan" på en fil med namn "kalle".

```
echo "Hejsan" > kalle
```

Om en fil redan finns och man vill lägga till text till den kan man använda sig av operatorn >> istället. Följande kommando lägger till ordet "Hejsan" i slutet av filen "kalle".

```
echo "Hejsan" >> kalle
```

3.1.3 Pipelines

En annan form av omdirigering är pipelines. Med hjälp av pipe (ett lodstreck |) kan man skicka utskriften från ett program som indata till ett annat (skicka stdout från program1 som stdin för program2). Följande kommando tar fram de rader i filen resultat1.txt som innehåller texten "son" men inte texten "Johan" (du kan läsa mer om grep i stycke 3.2).

```
grep son resultat1.txt | grep -v Johan
```

3.1.4 Processsubstitution

Vissa program vill ha filer som indata men vad ska man göra om filen behöver behandlas först? Antingen gör man jobbet i flera steg eller använder man sig av processsubstitution. Följande kommando skriver "Hello world" separerat med ett tab-tecken (du kan läsa mer om `paste` i stycke 3.2).

```
paste <(echo 'Hello') <(echo 'world!')
```

3.2 Några bra kommandon

Här följer en lista på några bra kommandon med korta beskrivningar. För mer utförlig information rekommenderas respektive programs manualsidor.

man Ett system för att läsa manualsidor för olika program. Tryck h för att se tillgängliga kommandon.

grep grep står ursprungligen för Global RegExp Print (dvs sök efter ett uttryck och skriv ut matchande rader). I grunden tar grep indata på formatet `grep sökord fil` och skriver ut alla rader i filen "fil" som innehåller ordet "sökord". Sökord kan vara ett reguljärt uttryck (se 3.3) för ökad användbarhet och grep kan även ta indata från stdin istället för en indatafil. Med väljaren `-v` matchar grep rader som inte innehåller sökordet.

cut cut klipper ut kolumner i en fil. Kolumnerna kan specificeras på flera olika sätt, antingen med en avgränsare (delimiter) med väljaren `-d` i kombination med väljaren `-f` för att välja en specifik kolumn eller ett visst antal tecken med väljaren `-c`. Se manualsidorna för specifikation av hur man väljer flera kolumner eller tecken.

paste paste slår ihop flera filer. Varje rad i filerna slås samman med motsvarande rad i övriga filer och separeras med ett tabulatortecken.

bc bc, eller basic calculator är en liten miniräknare man kan använda i terminalen. Kallas även bench calculator.

dc dc, eller desk calculator är även det en miniräknare. Skillnaden mot bc är att dc jobbar i postfixformat vilket gör att den oftast är lättare att använda i script. För att dc ska skriva ut resultatet måste man ge kommandot p (som i print). För att låta dc beräkna 5*4 kan man skriva följande: `echo "4 5 * p" | dc`. Om du startat dc och vill avsluta trycker du q och enter.

sed sed, eller stream editor är ett verktyg som kan modifiera en text. sed är väldigt praktiskt att använda sig av då det kan göra mycket. Ett exempel på vanlig användning är att byta ut en text mot en annan. `sed 's/text1/text2/g' fil` byter ut alla förekomster av "text1" mot "text2" i filen "fil" och skriver ut resultatet. Utbytestexten "text1" kan vara ett reguljärt uttryck. Om man vill skriva över ursprungsfilen kan man använda sig av väljaren `-i` (in-place). Istället för en indatafil kan sed även ta emot texten från stdin.

cat cat slår samman filer och skriver ut resultatet på stdout. Den vanligaste användingen av cat är dock att skriva ut innehållet av en fil då cat kan ta ett eller flera argument.

less less används även det för att läsa textfiler i terminalen. Fördelen med less är att det t.ex. har stöd för att gå igenom filen med piltangenterna. Man kan även stega flera rader (en skärmbild) med mellanslagstangenten.

sort sort sorterar rader. Normalt sätt sorteras raderna alfabetiskt i stigande ordning. Väljaren `-n` sorterar numeriskt och `-r` sorterar i sjunkande ordning.

3.3 Reguljära uttryck

Reguljära uttryck (regular expressions, regexp) är ett litet språk i sig som används för att matcha ett mönster mot text. I detta stycke ges en kort förklaring då det är bra att ha i det dagliga terminalarbetet. Ämnet behandlas noggrannare på en senare föreläsning.

Reguljära uttryck används väldigt flitigt inom UNIX-världen, till exempel kan du söka efter filer, anropa program med mönster och mycket mer. Normalt sett matchar ett tecken mot sig själv. En enkel punkt matchar valfritt tecken utom slut på raden och hakparenteser [] kan användas för att ge olika alternativ. Vanliga parenteser betyder en grupp av tecken. Dessutom finns det fyra olika sätt att visa på antal matchningar `+ * ?` och måsvingar . Plus och asterisk används för att matcha 1 eller flera respektive noll eller flera gånger. Med frågetecknet kan man visa att föregående tecken inte krävs. Måsvingarna specificerar ett visst antal (eller ett visst intervall) förekomster. Om man vill matcha ett specialtecken kan man använda sig av backslash. Två specialtecken som kan vara bra att ha är ^ och $ som matchar början respektive slutet av en rad.

Nedan följer några exempel

Mönster	Matchar	Matchar inte
abc	abc	ab
abc?	abc,ab	ac
ab+c	abc, abbc	ac
ab*c	ac, abc, abbc	a, c
a[bc]d	abd, acd	ad, abcd
a(bc)?d	ad, abcd	abd
a(bc){2}d	abcbcd	abcd, ad
ab{1,2}c	abc, abbc	ac, abbbc
a\.b	a.b	abb
^ab$	ab	abc